Japanese Using JMock with Groovy

JMock is a popular mocking framework for Java. Several versions are available.

The sections below illustrate using various versions of JMock for the mocking parts of Using Testing Frameworks with Groovy.

The Item Storer Example

We are going to consider how you might use JMock as part of testing the Item Storer Example.

Here is how we can test `JavaStorer` using version 1.x of JMock using its JUnit 3 integration:

```
// require(groupId:'jmock', artifactId:'jmock-core', version='1.2.0')
// require(groupId:'junit', artifactId:'junit', version='3.8.2')
import org.jmock.MockObjectTestCase

class JMock1Test extends MockObjectTestCase {
    def mockControl, mockReverser, storer

    protected void setUp() throws Exception {
        mockControl = mock(Reverser.class)
        mockReverser = mockControl.proxy()
        storer = new JavaStorer(mockReverser)
    }

    void testStorage() {
        expectReverse(123.456, -123.456)
        expectReverse('hello', 'olleh')
        checkReverse(123.456, -123.456)
        checkReverse('hello', 'olleh')
    }

    def expectReverse(input, output) {
        // with is a keyword in Groovy so we quote it

mockControl.expects(once()).method('reverse').'with'(eq(input)).will(returnValue(outpu
t))
    }

    def checkReverse(value, reverseValue) {
        storer.put(value)
        assert value == storer.get()
        assert reverseValue == storer.getReverse()
    }
}

def suite = new junit.framework.TestSuite()
suite.addTestSuite(JMock1Test.class)
junit.textui.TestRunner.run(suite)
```

Here is how we can test `JavaStorer` using version 2.x of JMock using its JUnit 4 integration:

```
// require(groupId:'junit', artifactId:'junit4', version='4.3.1')
// require(groupId:'org.jmock', artifactId:'jmock', version='2.1.0')
// require(groupId:'org.jmock', artifactId:'jmock-junit4', version='2.1.0')
import org.jmock.integration.junit4.JMock
import org.jmock.Mockery
import org.junit.Test
import org.junit.Before
import org.junit.runner.RunWith
import org.junit.runner.JUnitCore

@RunWith(JMock)
class JMock2Test {
    Mockery context = new JUnit4GroovyMockery()
    def mockReverser, storer

    @Before void setUp() throws Exception {
        mockReverser = context.mock(Reverser.class)
        storer = new JavaStorer(mockReverser)
    }

    @Test void testStorage() {
        expectReverse(123.456, -123.456)
        expectReverse('hello', 'olleh')
        checkReverse(123.456, -123.456)
        checkReverse('hello', 'olleh')
    }

    def expectReverse(input, output) {
        context.checking{
            one(mockReverser).reverse(input); will(returnValue(output))
        }
    }

    def checkReverse(value, reverseValue) {
        storer.put(value)
        assert value == storer.get()
        assert reverseValue == storer.getReverse()
    }
}

JUnitCore.main('JMock2Test')
```

To make our tests a little more DSL-like, we used the following helper class with JMock 2:

```
import groovy.lang.Closure;
import org.jmock.Expectations;
import org.jmock.integration.junit4.JUnit4Mockery;

public class JUnit4GroovyMockery extends JUnit4Mockery {
    class ClosureExpectations extends Expectations {
        void closureInit(Closure cl, Object delegate) {
            cl.setDelegate(delegate);
            cl.call();
        }
    }

    public void checking(Closure c) {
        ClosureExpectations expectations = new ClosureExpectations();
        expectations.closureInit(c, expectations);
        super.checking(expectations);
    }
}
```